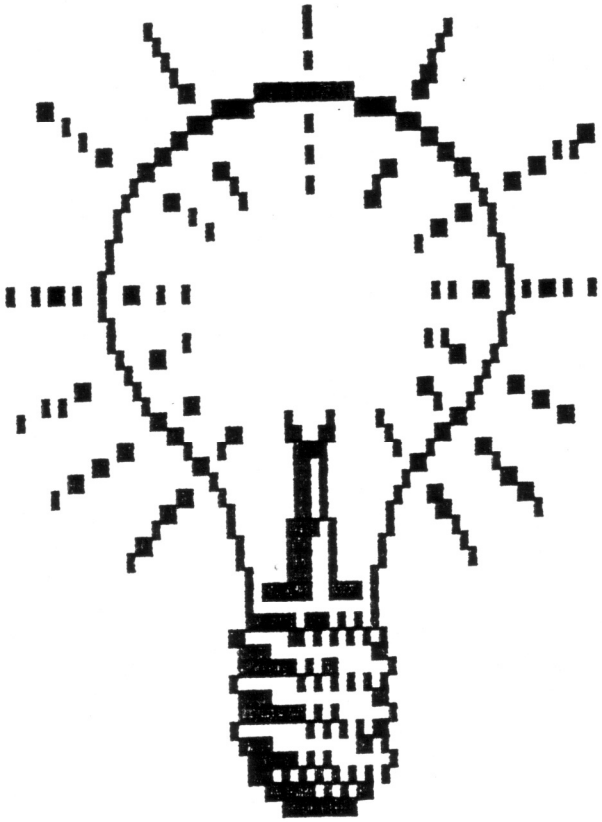
VZ-LINK MAGAZINE

ISSUE No. 28 JANUARY 89

NEW ZEALAND

VZ 300/200

COMPUTER USERS CLUB



NEWS AND VIEWS

I hope you have had a good christmas break and are settling into the new year with ease.
I am sorry this issue is very late,but thing's have been rather busy at work.Anyway we made it and that's what counts.

The cover is just a little hint to remind you that we are still looking for IDEA's for a front cover.
If you have some drawing's please feel free to send them in. I will try and redraw them for you.

I hope you enjoy this issue of LINK .I hope There is something for everyone.
If your Sub's are due please send them in as soon as possible.

NOTE

I have once again had to change the date for the meeting.The new date and details are below.
I hope you can make it and don't forget your software etc.

STRUCTURED PROGRAMMING

STRUCTURED PROGRAMMING - Part 1 by Chris Hobrough.

Articles in computer magazines over the last couple of years have frequently referred to the idea of structured programming.
Languages such as Pascal or C are said to be structured while BASIC is criticized for its lack of structure.
So what is structured programming and why do we need it?

The structured approach to programming involves a number of ideas and affects everything from the first stages of design to the presentation of the finished product. It is an approach which makes programs easier to design and debug and easier to modify later on. In addition, programs are more readable and understandable.
There are three main ideas which are central to structured programming : top-down design, a modular approach and the use of formal command structures within modules. In this article I plan to illustrate each of these concepts by looking at some of the steps involved in designing a program. In addition, I'll be discussing the importance of adequate documentation and clear presentation of the finished program.
I don't intend looking too deeply into the programming details and I'll be using pseudo-code some of the time. Pseudo-code, for those not familiar with it, is simply the use of short, plain Engish statements to describe what the program is doing. I find it more natural than flow charting, though I sometimes use both. Once a program is designed in pseudo-code, it can then be translated into any programming language.

Top-down Design & Modularity

Designing a program is essentially a matter of solving a problem and the first step in solving any problem is to define precisely what is needed. What do we want our program to do? Top-down design implies that we start with the broadest view and gradually refine it until we get to the details.

The program I'll be using to illustrate these techniques is a dedicated data base manager for a video library.
The data base will consist of records held in a disk file with each record containing the title of a video together with certain information relating to that video. The program will have to access this data file, add or delete titles, search for a particular title and list all the titles in the file.

Programs of this type tend to have a similar starting point as far as the user is concerned. First, a menu is displayed from which we select an option. After the selection is made we go to one of the deeper levels of activity, perhaps another menu, and perform the operations we want. When finished, we return to the main menu for another option. At some point we'll select the option to quit the program and the session will end.

To implement this we obviously need a loop of some kind. Inside the loop will be the menu options and some routines to process our selection. The loop needs an escape route, so we can get out of it, and there will be some things to do both before and after the loop. Without going any deeper we can write an outline of the program in pseudo-code :-

```
Initialize                          (sub-routine)
Main menu                           (  "     "   )
While choice is not "EXIT"
     Process choice                 (  "     "   )
     Main menu                      (  "     "   )
Next
Terminate                           (  "     "   )
End
```

The first thing you'll notice is that the whole program has been reduced to a series of sub-routines or modules and that each routine represents a heading for a major step in the program. The loop is controlled by the WHILE ... NEXT structure. This is a command structure which is available with most modern versions of BASIC and can be easily simulated on the VZ. The WHILE loop is basically saying that, if the user has not selected the exit option from the menu, go into the body of the loop and process the user's choice, then dispay the menu again for another selection. When the next selection is made, execution loops back to the WHILE statement and the test is made again. If the exit option is chosen then we jump the loop and finish the session. In order for the WHILE to have something to test the first time, the menu routine is called once before entering the loop.

Most programs have a number of housekeeping chores which need to be done before they get going so we group them all under the heading "Initialize", even though we may not know what they'll be yet. In our program an array will be set up to hold the records and the data will have to be read into it from the disk file. After quitting the program, any records which have been altered, added or deleted will have to be written back to disk and this can be done in the routine called "Terminate".

What we have so far is the main control module. The next step is to look at each sub-routine or module in turn and go through a similar process of splitting it up into its own sub-modules until finally we get to the point where each line can be represented by a single statement in BASIC, or very close to it. In so doing we build up a sort of hierarchy of ever more detailed modules, each of which is designed and debugged independantly. If we find a problem, we can be pretty sure it's in the module we're working on at the moment because the others have already been debugged.

Before we go on let's translate what we have so far into BASIC. The main menu routine will return the user's selection in the variable called CHoice$.

```
1000 ' Main Control Module .... ..............................
1002 '
1010 GOSUB 2000                  ' Initialize
1030 GOSUB 3000                  ' Main menu
1040 WHILE CHoice$ <> "Exit"     ' While choice not "Exit"
1050     GOSUB 4000              '     Process choice
1060     GOSUB 3000              '     Main menu
1070 NEXT                        ' Next
1080 GOSUB 5000                  ' Terminate
1090 END                         ' End
1998 '
2000 ' Initialize ............. ..............................
2002 '
2990 RETURN
2998 '
3000 ' Main Menu .............. ..............................
3002 '
3990 RETURN
3998 '
4000 ' Process Choice ......... ..............................
4002 '
4990 RETURN
4998 '
5000 ' Terminate .............. ..............................
5002 '
5990 RETURN
```

With the exception of the WHILE statement, which we haven't dealt with yet, you could enter this and it would run. It won't do much yet but we have the main module and the headings for the first four sub-routines. Note that the pseudo-code is simply moved across and becomes the REMarks. You can see that this framework, perhaps with some minor modifications, could be used for a wide variety of applications. So far, the only specific thing in it is a menu of some sort. The important point is that we've got over that initial step, of knowing where to start, by applying a simple technique.

In Part 2 of this article we'll look at some of the sophisticated command structures that appear in modern versions of BASIC and see how they can be simulated on the VZ. At the same time we'll expand our data base program and examine what happens within some of the modules.

PERIPHERALS FOR THE VZ200/300

Part 1. PRINTERS

By P Richardson.

A printer is probably one of the most useful add-ons you can buy for your computer. Before we look at what it can do for you we will see what is available to the VZ user.

The majority of printers fall into 1 of 4 categories. These are:-

1. Thermal Printers
2. Dot Matrix Printers
3. Daisy Wheel Printers
4. Laser Printers

Thermal Printers

With a Thermal Printer the character to be printed is "burned" into the paper by the print head. This requires the use of specially treated paper, which is a cost disadvantage.

Dot Matrix Printers

In a Dot Matrix Printer the print head consists of (usually) 9 vertically alligned pins, each of which is able to "shoot" a dot of ink from the ribbon onto the paper. The appropriate pins are fired as the print head traverses across the paper.

e.g.:-

```
Pin No.

1.        *       *           * * * *
2.        *       *                 *
3.        *       *               *
4.         *     *              *
5.          *   *              *
6.           * *              *
7.            *               * * * *
8.
9.      ********************************
```

As can be seen from the above diagram, each pin is fired individually to produce the requred charcter. Note that only seven pins are used to produce a character and that in the example above pin 9 was used to produce an underline. Since each pin can be fired seperately, user defined characters sets (e.g italics, French etc.) & graphics shapes can also be produced.

Daisy Wheel Printers

Daisy Wheel Printers produce characters using a method similar to a typewriter. A wheel with the characters embossed on it is pushed against the paper with the ribbon in between to print the required character. The print quality of a Daisy Wheel Printer approaches that of a good typewriter.

Laser Printers

A Laser Printer is a high speed, high quality device that uses laser beams to write characters onto an electrically charged surface. The charcters are formed from a matrix of many densely packed dots, and are then beamed onto a photographic surface. This creates a character pattern with an electrical charge.

The charged pattern attracts a toner, thus developing' the character image. The image is transferred to paper by heat or pressure, and fixed there by passing the paper through a chemical vapour bath. This process is very similar to that used in many photocopying machines. Laser printers have full graphics and colour capabilities, and can produce many different typestyles.

Obviously the more you spend on a printer, the better the output quality you will get. For most of us poor peasants, a Dot Matrix printer will do very nicely. The Daisy Wheel is really only justifiable, on a cost basis, if you are printing a reasonable amount of high quality correspondence.

Almost all printers are capable of being driven by 'software control'. This allows the user to create very professional looking documents with things such as **expanded text**, condensed text, <u>underline</u> and a whole host of other features.

The way to access these facilities is outlined in your printers' manual. Most manuals give the required codes and usually a sample listing to demonstrate the effect. The codes are often called 'escape codes' and are written in the form of "ESC" plus one or two additional characters as required.

e.g.

"ESC + E"

These codes are entered as CHR$ codes. Thus the example on the previous pagfe would be typed into a program as;

(Line No.) LPRINT CHR$(27);CHR$(69);

Note the semi-colons between the CHR$ codes and at the end of the line. This ensures a CR/LF (carriage return/line feed) is not generated by the VZ as this would stop the code having any effect on the following text.

The moral of the story is "read & understand your printer manual and you can print some very readable documents with ease."

FOR SALE - SCREEN ED - FOR SALE

VZ SCREEN ED - $25.00 + $2.00 POST & HANDLING.
VZ SCREEN ED IS A NEW DRAWING PROGRAM FOR HI-RES SCREENS WITH MANY POWERFUL OPTIONS. BOTH TAPE AND DISK USERS ARE CATERED FOR.

SEND YOUR CHEQUE OR MONEY ORDER TO :-
MATTHEW TAYLOR SOFTWARE -- OR PHONE (049) 75 2350
38 FISHING POINT ROAD RATHMINES NSW 2283 AUSTRALIA

YOU CAN SEE MATTHEW AT CLUB MEETINGS FOR A DEMONSTRATION OR PURCHASE OF SCREEN ED.

HINTS & TIPS . . . . . .

I DON'T AS A RULE SPEND MUCH TIME PROGRAMMING AND RECENTLY SPENT COUPLE DAYS AT IT. THIS PARTICULAR PROGRAM HAD ME STYMED FOR MONTHS AND THE ANSWER WAS STARING ME IN THE FACE AND I COULD'NT SEE IT, IE :-

```
30 DIM AR(12,18):DIM A(18)
40 CLEAR 2000:GOSUB 2000
```

THE DEFAULT FOR ARRAYS IS 0-10. THE CLEAR COMMAND IN LINE 40 REDUCED THE ARRAYS IN LINE 30 TO THE DEFAULT VALUE. THE CLEAR COMMAND MUST BE USED BEFORE DIMENSIONING.

P.O.BOX 1972 C.P.O AUCKLAND N.Z

(C) AUCKLAND VZ 300/200 COMPUTER USER'S CLUB 1986

ALL MATERIAL IN THIS NEWSLETTER IS SUBJECT TO COPYRIGHT..
CONTRIBUTED PROGRAMS & ARTICLES CAN BE REPRODUCED ON THE UNDERSTANDING THAT THEY ARE FOR THE PRIVATE USE OF SUBSCRIBERS ONLY.
COPYRIGHT IS ALWAY'S RETAINED BY THE AUTHOR.
THIS CLUB WILL NOT BE HELD RESPONSIBLE FOR THE BREECHING OF COPYRIGHT CAUSED BY THE PUBLICATION OF ANY MATERIAL SENT IN UNDER A FALSE NAME.

YOU ARE WELCOME TO PHONE ME

SUITE BY ROBERT QUINN . . . .

NOTES FOR PUTGET*3

PUT and GET work from within a BASIC program, using statements based on token 165 for PUT and token 164 for GET. These tokens have to be emplaced in a program, as they are in PUTGET*3 where 'P' and 'G' characters are used to reserve bytes in lines 16 and 18 for the PUT and GET tokens. The number characters '1' and '2' in lines 16 and 18 (following 'P' and 'G') modify the PUT and GET statements to make them PUT1, PUT2, GET1, GET2 statements.

When PUTGET*3 is RUN, you will find that most of lines 16 and 18 are invisible when listed, because your VZ is unable to display the words PUT and GET. The words do not exist for the VZ, only the tokens.

DO NOT TYPE IN the REM lines at the start of PUTGET*3 when you enter the program on your VZ, otherwise RUNing the program will POKE the tokens to the wrong locations and the program won't work.

Line 20 can be rewritten as follows to exclude a RETURN from the subroutine (GOSUB12 or GOSUB14) when the RETURN key is pressed :-

```
20 IFK$=CHR$(13)OR(K$>0ANDK$<7)THEN14ELSERETURN
```

Both PUTGET*3 and INTDATA (see issue 17 of the JOURNAL) set up their routines in the same part of the COMMUNICATIONS REGION, so you cannot have both routines available simultaneously in your VZ.

SUITE is a combined PUTGET*3 and modified INTDATA for disk drive owners, that allows you to use both sets of routines. SUITE sets up and activates the INTERRUPT ROUTINE and four associated subroutines in protected high memory, leaving the COM.REGION location to PUTGET*3. Three of the four subroutines to the INTERRUPT are those of INTDATA :-

<SHIFT> <C> executes CURSOR CLEAR SCREEN.
<SHIFT> <V> executes LDIR:LSTATUS
<SHIFT> <B> executes DIR:STATUS

<SHIFT> <X> now executes a new function (IBS) that allows you to BSAVE screens to disk whenever you want.

IBS saves the whole 2K of video memory. In MODE(1) this will be a single 2K hi res screen. In MODE(0) it will be a single 512 byte text screen and the content of the remainder of video memory. This remainder can be used in MODE(0) to store three text screens using PUT, PUT1 and PUT2 within a BASIC program. Then <SHIFT> <X> will save the current screen and all three stored screens to disk as a single binary file with the file name SCREEN00.

You can use <SHIFT> <X> again and again to save the current content of video memory to disk. Each time, the number at the end of the file name will increment so that your files will save under the file names SCREEN01, SCREEN02, SCREEN03, ..., SCREEN99 then (if you should ever be busy enough with your VZ and disk drive to get this far!) back to SCREEN00.

SUITE CONT. . . . . .

When you BLOAD a SCREEN file that consists of four MODE(0) screens, you can use GET, GET1 and GET2 to recall the three alternate screens to display.

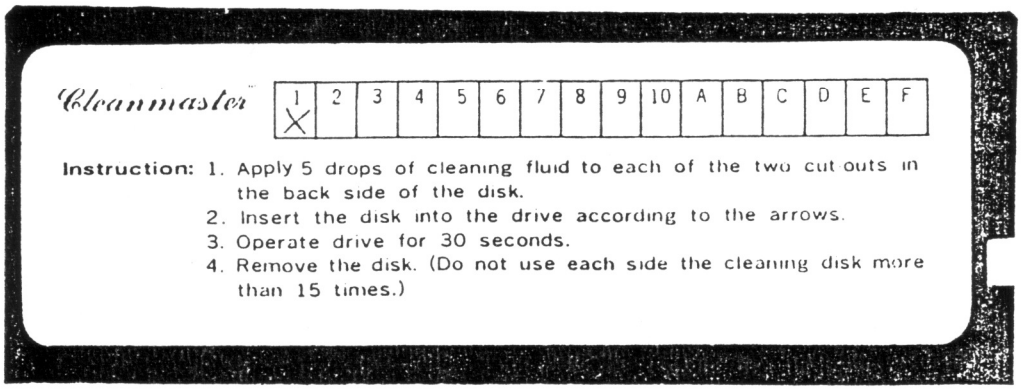
To use PUT and GET you will need a subroutine in your programs like the one at the start of PUTGET*3: lines 10 to 24 (Issue 19, page 4) .

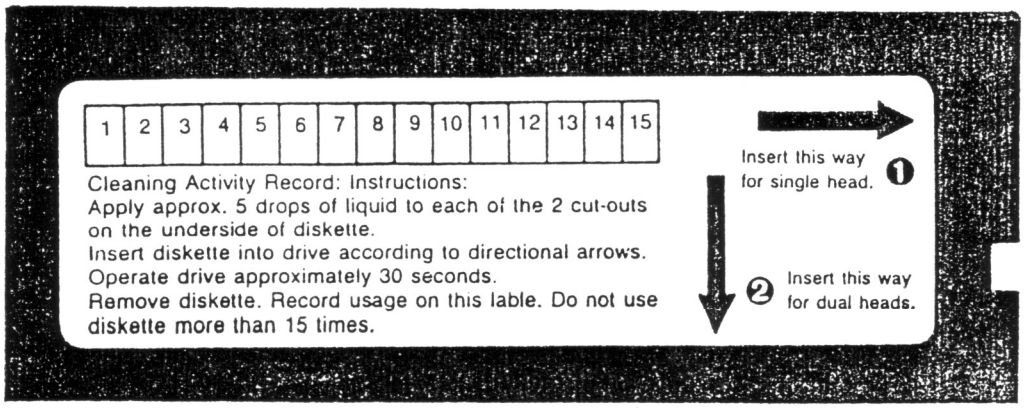
Immediately after BLOADing a SCREEN file, the INTERRUPT is disabled and will need to be re-enabled before <SHIFT> <X>, <C>, <V>, <B> can be used again. Re-enabling happens automatically in command mode (when a BASIC program is not running) and when a BASIC program is BREAKed, STOPed or ENDed.

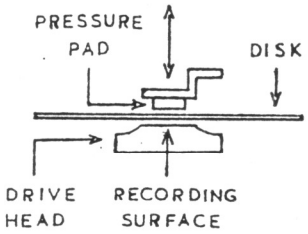
The most convenient way to do it without stopping a running program is to execute a SOUND statement from within your program, immediately following the BLOAD statement in the program. If you don't want a noise then use SOUND0,1.

```
5 FORR=1TO217:READA:B=B+A:NEXT
10 IFB<>18830THENPRINT"[illegible]":SOUND30,2:END
14 REM"[illegible]
15 A=(PEEK(30897)+PEEK(30898)*256)-144::B%=A/256:C%=A-B%*256
20 POKE30897,C%:POKE30898,B%:CLEAR50
30 A=PEEK(30897)+PEEK(30898)*256:E=65536:B%=31273
40 FORR=1TO73:READC%:POKEB%,C%:B%=B%+1:NEXT
50 D=A+1-E:FORR=74TO217:READC%:POKED,C%:D=D+1:NEXT
60 D=A+21:B%=D/256:C%=D-B%*256:POKEA+43-E,C%:POKEA+44-E,B%
70 D=A+29:B%=D/256:C%=D-B%*256:POKEA+49-E,C%:POKEA+50-E,B%
80 D=A+1:B%=D/256:C%=D-B%*256:POKE30846,C%:POKE30847,B%
90 POKE30845,195:POKE31107,41:POKE31108,122:POKE31104,72
95 POKE31105,122
99 REM"[illegible]
100 DATA17,0,114
110 DATA183,40,19,254,58,40,15,35,254,49,17,0,116,40,7,254,50
120 DATA17,0,118,32,46,229,33,0,112,24,31,229,33,0,114,17,0,112
130 DATA183,40,21,254,58,40,17,225,35,229,254,49,33,0,116,40
140 DATA7,254,50,33,0,118,32,7,1,0,2,237,176,225,201,225,195
150 DATA151,25
160 REM"[illegible]
170 DATA58,251,104,254,115,40,70,254,121,40,30,254,91,40,93,254
180 DATA122,40,113,201
189 REM"[illegible]
190 DATA34,83,67,82,69,69
200 DATA78,48,48,34,44,55,48,48,48,44,55,55,70,70,0
209 REM"[illegible]
210 DATA33,0,0,205,57,72,33,0,0,126,254,57,48,3,60,119,201,62
220 DATA48,119,43,126,254,57,56,244,62,48,119,201
225 DATA0,0,0,0,0,0
229 REM"[illegible]
230 DATA42,32,120,1,0,114,58,24,120,254,0,62,32,32,2,62,96,119
240 DATA50,60,120,35,121,189,32,236,120,188,32,232,201
249 REM"[illegible]
250 DATA205,196,5,203,71,32,7,62,1,50,156,120,24,15
270 DATA33,230,0,1,75,0,205,92,52,201
279 REM"[illegible]
280 DATA62,0,50,156,120,205,6,73,205,205,82,201
```

HEAD CLEANING DISK UPDATE . .







SINCE MY MISHAP WITH THE CLEANMASTER HEAD CLEANING DISK EARLIER IN THE YEAR I'M PLEASED TO ANOUNCE THAT AFTER BRINGING IT TO THE ATTENTION OF D.SMITH THEY HAVE RECTIFIED THE PROBLEM BY PLACING A LABEL ON THE TOP CUTOUT WITH INSTRUCTIONS TO REMOVE FOR DUAL HEAD DRIVES. ALSO MY DAMAGED DRIVE WAS FIXED AT NO CHARGE.

NOT LONG AGO I'VE COME ACROSS A SIMILAR H.C/DISK WHICH IS CORRECTLY LABELLED, REFER TO THE TWO LABELS FOR COMPARISON. FOR THOSE PERSONS NOT AWARE THE VZ AND C64 WRITE TO THE BOTTOM OF DISK, NOT THE TOP. THE REASON FOR THE CUTOUT AT TOP OF DISK IS SO THE PRESSURE PAD CAN REST ON DISK SURFACE BRINGING IT INTO CONSTANT CONTACT WITH RECORDING HEAD WHICH IS AT BOTTOM.

DUAL HEAD DRIVES HAVE 2'ND HEAD IN PLACE OF PRESSURE PAD SO THEY CAN WRITE TO BOTH SIDES OF DISK WITHOUT FLIPPING OVER.

BOTH C/DISKS MUST BE INSERTED INTO DRIVE AS DEPICTED ON BOTTOM LEFT AND RIGHT. IF YOU HAVE DIFFERENT H.C/DISK OR ABOVE MAKE SURE THAT AT NO TIME PRESSURE PAD RESTS ON ANY PART OF HEAD CLEANING SURFACE AS DAMAGE MAY RESULT.

BIT - BY - BIT

This month's routine is another example of an excellent contribution by a reader - Mr Bruce Harris of Malabar NSW. It's great to see that someone is writing material for the VZ which is really useful and inovative. The program listed here allows the VZ to be used as a Frequency Counter, via the cassette input port. It is in BASIC, so that everyone can use it, and it loads five machine code routines into memory. These are displayed as they are loaded. The screen then displays the frequency in large graphic numerals in the centre of the screen. The left-hand top characters show the count time in hex, the middle two the frequency, and the right number is the count rate. These could be moved to another place in memory if required. The count time can be altered in the BASIC program, or 'on line' by pressing a few keys. This allows the counter to be calibrated against a known frequency.

The program can be demonstrated just by playing a cassette tape into it. Unfortunately, the frequency response of the tape input does not seem to be too good, and counts above 5000 or 6000 Hz are not accurate. An input through a port should be able to cover beyond 20,000Hz, but this has not been tried.

Bruce will supply a copy of the program on tape if you can't be bothered typing it in, together with the Assembly listing and comments for $12 post paid, or in return for a USEFUL SIGNIFICANT program (NOT a game). His address is 8 Howe St, MALABAR 2036. Thanks very much Bruce!

```
1 CLEAR 100:COLOR,1
2 S=8*16^3+9*16^2-16^4
3 GOSUB 100
4 S=8*16^3+10*16^2-16^4
5 GOSUB 100
6 S=8*16^3+10*16^2+32-16^4
7 GOSUB 100
9 S=8*16^3+10*16^2+6*16-16^4
10 GOSUB 100
11 S=8*16^3+10*16^2+7*16+8-16^4
12 GOSUB 100
13 S=8*16^3+10*16^2+11*16-16^4
14 GOSUB 100
17 POKE 30744,1
18 CLS
19 GOSUB 1000
20 Q=7*16^3+9
22 POKE Q,164:POKE Q+1,171
80 POKE 30862,00:POKE 30863,137
85 X=USR(X)
90 POKE 30744,0
95 END
100 REM ENTER M/L INTO MEMORY
130 I=0
140 N=0
150 READ A$
151 IF A$="END" THEN PRINT
```

(... cont'd)

```
152 PRINT A$;
153 IF A$="END" THEN PRINT
155 IF A$="END" THEN RETURN
160 FOR J=1 TO 2
170 X=ASC(MID$(A$,J))
180 N=16*N+(X AND 15)-9*(X>64)
190 NEXT J
200 POKE S+I,N
210 I=I+1
215 GOTO 140
220 END
500 DATAF3,F5,C5,D5,E5,CD,00,8A
510 DATACD,20,8A,CD,60,8A,CD,F4
520 DATA2E,B7,28,F1,2A,08,70,FE
530 DATA31,28,21,FE,21,28,26,FE
540 DATA32,28,2B,FE,22,28,30,FE
550 DATA33,28,35,FE,23,28,3A,FE
560 DATA34,28,3F,E1,D1,C1,F1,FB
570 DATAC9,00,00,00,01,01,00,09
580 DATA22,08,70,18,BE,01,FF,FF
590 DATA09,22,08,70,18,B5,01,10
600 DATA00,09,22,08,70,18,AC,01
610 DATAF0,FF,09,22,08,70,18,A3
620 DATA01,00,01,09,22,08,70,18
630 DATA9A,01,00,FF,09,22,08,70
640 DATA18,91,01,00,10,09,22,08
650 DATA70,18,88,00
670 DATA "END"
700 DATAED,5B,08,70,21,00,00,3A
702 DATA00,68,E6,40,B8,01,00,00
704 DATACB,11,09,47,1B,7A,B3,20
706 DATAEE,22,10,70,C9,00,00,00
708 DATA "END"
710 DATA2A,10,70,11,16,70,01,10
712 DATA27,CD,48,8A,01,E8,03,CD
714 DATA48,8A,01,64,00,CD,48,8A
716 DATA01,0A,00,CD,48,8A,01,01
718 DATA00,CD,48,8A,C9,00,00,00
720 DATA3E,FF,3C,A7,ED,42,30,FA
722 DATA09,C6,30,12,13,C9,00,00
724 DATA "END"
730 DATA21,16,70,11,61,70,06,05
732 DATACD,78,8A,7B,C6,06,5F,10
734 DATAF7,C9,00,00
736 DATA "END"
740 DATAD5,C5,7E,D6,30,87,87,4F
742 DATA87,87,81,06,00,4F,E5,21
746 DATAB0,8A,09,06,05,C5,01,04
748 DATA00,ED,B0,7B,C6,1C,5F,C1
750 DATA10,F3,E1,23,C1,D1,C9,00
752 DATA "END"
800 DATA91,9C,9C,99,95,90,91,9D
805 DATA95,91,98,95,95,98,90,95
810 DATA90,9C,9C,98
815 DATA90,91,9A,90,90,98,9A,90
820 DATA90,90,9A,90,90,90,9A,90
825 DATA90,9C,9C,98
830 DATA91,9C,9C,99,90,90,90,96
835 DATA90,96,9C,90,95,90,90,90
845 DATA94,9C,9C,9C
855 DATA91,9C,9C,99,90,90,90,95
865 DATA90,94,9C,99,91,90,90,95
875 DATA90,9C,9C,98
885 DATA90,90,91,9A,90,91,98,9A
895 DATA91,98,90,9A,94,9C,9C,9E
897 DATA90,90,90,98
900 DATA95,9C,9C,9C,95,93,93,90
905 DATA90,90,90,99,91,90,90,96
910 DATA90,9C,9C,90
915 DATA90,96,9C,98,95,90,90,90
920 DATA95,9C,9C,99,95,90,90,95
925 DATA90,9C,9C,98
930 DATA95,9C,9C,9D,90,90,91,98
935 DATA90,91,98,90,90,95,90,90
940 DATA90,94,90,90
945 DATA91,9C,9C,99,95,90,90,95
950 DATA91,9C,9C,99,95,90,90,95
955 DATA90,9C,9C,98
960 DATA91,9C,9C,99,95,90,90,95
970 DATA90,9C,9C,9D,90,90,90,96
975 DATA90,9C,9C,90
999 DATA "END"
1000 RETURN
```

** NEW *** FOR SALE *** NEW **

E & F WORD PROCESSOR PATCH 3.3

THIS LATEST PATCH WRITTEN BY DAVE MITCHELL WILL CONVERT YOUR E & F TAPE WORD PROCESSOR FOR FULL DISK USE WHILE RETAINING ALL TAPE FUNCTIONS. BELOW ARE THE DISK COMMANDS & ADDED FUNCTIONS :-

LOAD, SAVE, ERASE, RENAME, DIRECTORY, INITIALIZE, UPDATE, DRIVE 1 & 2, SHIFTLOCK & IMBEDDED PRINTER CONTROL CODES

PATCH 3.3 HAS PROVISION FOR IMBEDDING PRINTER CONTROL CODES IN TEXT AND FAST SAVING AND LOADING OF TEXT DATA TO AND FROM DISK USING BLOCK SAVE/LOAD TECHNIQUES.

MINIMUM MEMORY REQUIREMENTS :-
VZ300 + 16K RAM PACK OR VZ200 + 18K (16K RAM PACK + 2K)

PATCH 3.3 IS COPYRIGHT TO AND ONLY AVAILABLE FROM :-
HUNTER VALLEY VZ USERS' GROUP P.O.BOX 161 JESMOND 2299
N.S.W. AUSTRALIA - Phone (049) 51 2756

PRICE - AUS/NZ AU$20.00 - UPDATE - AUS-$10.00 - NZ-AU$11.00.
UPDATING AVAILABLE ONLY TO PREVIOUS PURCHASERS OF OUR PATCHES.

FOR MORE INFORMATION WRITE TO H.V.VZ.U.G. ENCLOSING A SSAE.

FOR PRIVATE SALE

EXTENDED DOS V1.3 -$15.00

THE PREVIOUS VERSION HAS BEEN UPDATED WITH EXTRA COMMANDS ADDED.

OLD COMMANDS - MERGE, DIRA, LDIRA, DIRB, LDIRB, OLD, OLD., DEC, HEX, STATUSA AND LSTATUSA. STATUSA AND LSTATUSA ALSO WORKS WITH VERSION 1.0 DOS.

NEW COMMANDS :-
MENU - LOADS AND RUNS BINARY OR TEXT MENU PROGRAM FROM DISK.
CODE - SIMPLIFIES USING PRINTER CONTROL CODES DIRECTLY OR FROM WITHIN A PROGRAM.
LTAB - IS FOR SETTING OF LEFT MARGIN.
MOVE - MOVES BASIC FILE FROM DISK TO CHOSEN MEMORY ADDRESS.
UPD - ERASES OLD FILE AND SAVES WITH SAME FILE NAME.

MENU/FILE COPIER/DISK ORGANIZER - $15.00

THIS UTILITY WILL READ YOUR DISK DIRECTORY AND PRESENT YOU WITH SEVERAL OPTIONS. USING THE CURSOR YOU CAN RUN/BRUN ANY PROGRAM OR SELECT FILE COPY, REN, ERASE, DRIVE 1 OR 2, ETC. BESIDES COPYING TEXT AND BINARY FILES ALL OTHER FILES CAN BE COPIED AS WELL EXEPT FOR DATA FILES.

FOR PURCHASE OR INFO CONTACT - DAVE MITCHELL - (079) 27 8519
24 ELPHINSTONE St. NORTH ROCKHAMPTON QUEENSLAND 4701

FOR INFORMATION OR DEMONSTRATION IN NEWCASTLE AREA CONTACT :-
JOE LEON - (049) 51 2756 - 22 DRURY ST. WALLSEND NSW 2287